Linux基础课

一、常用命令介绍

(1) ctrl c: 取消命令，并且换行
(2) ctrl u: 清空本行命令
(3) tab键：可以补全命令和文件名，如果补全不了快速按两下tab键，可以显示备选选项
(4) ls: 列出当前目录下所有文件，蓝色的是文件夹，白色的是普通文件，绿色的是可执行文件
(5) pwd: 显示当前路径
(6) cd XXX: 进入XXX目录下，cd .. 返回上层目录
(7) cp XXX YYY: 将XXX文件复制成YYY，XXX和YYY可以是一个路径，比如../dir_c/a.txt，表示上层目录下的dir_c文件夹下的文件a.txt
(8) mkdir XXX: 创建目录XXX
(9) rm XXX: 删除普通文件； rm XXX -r: 删除文件夹
(10) mv XXX YYY: 将XXX文件移动到YYY，和cp命令一样，XXX和YYY可以是一个路径；重命名也是用这个命令
(11) touch XXX: 创建一个文件
(12) cat XXX: 展示文件XXX中的内容
(13) 复制文本
    windows/Linux下：Ctrl + insert，Mac下：command + c
(14) 粘贴文本
    windows/Linux下：Shift + insert，Mac下：command + v

二、tmux教程

功能：
    (1) 分屏。
    (2) 允许断开Terminal连接后，继续运行进程。
结构：
    一个tmux可以包含多个session，一个session可以包含多个window，一个window可以包含多个pane。
    实例：

```
tmux:
    session 0:
        window 0:
            pane 0
            pane 1
            pane 2
            ...
        window 1
        window 2
        ...
    session 1
    session 2
    ...
```

操作：
    (1) tmux：新建一个session，其中包含一个window，window中包含一个pane，pane里打开了一个shell对话框。
    (2) 按下Ctrl + b后手指松开，然后按%：将当前pane左右平分成两个pane。
    (3) 按下Ctrl + b后手指松开，然后按"（注意是双引号"）：将当前pane上下平分成两个pane。

(4) Ctrl + d：关闭当前pane；如果当前window的所有pane均已关闭，则自动关闭window；如果当前session的所有window均已关闭，则自动关闭session。

(5) 鼠标点击可以选pane。

(6) 按下ctrl + b后手指松开，然后按方向键：选择相邻的pane。

(7) 鼠标拖动pane之间的分割线，可以调整分割线的位置。

(8) 按住ctrl + b的同时按方向键，可以调整pane之间分割线的位置。

(9) 按下ctrl + b后手指松开，然后按z：将当前pane全屏/取消全屏。

(10) 按下ctrl + b后手指松开，然后按d：挂起当前session。

(11) tmux a：打开之前挂起的session。

(12) 按下ctrl + b后手指松开，然后按s：选择其它session。

方向键 -- 上：选择上一项 session/window/pane

方向键 -- 下：选择下一项 session/window/pane

方向键 -- 右：展开当前项 session/window

方向键 -- 左：闭合当前项 session/window

(13) 按下Ctrl + b后手指松开，然后按c：在当前session中创建一个新的window。

(14) 按下Ctrl + b后手指松开，然后按w：选择其他window，操作方法与(12)完全相同。

(15) 按下Ctrl + b后手指松开，然后按PageUp：翻阅当前pane内的内容。

(16) 鼠标滚轮：翻阅当前pane内的内容。

(17) 在tmux中选中文本时，需要按住shift键。（仅支持Windows和Linux，不支持Mac，不过该操作并不是必须的，因此影响不大）

(18) tmux中复制/粘贴文本的通用方式：

(1) 按下Ctrl + b后松开手指，然后按[

(2) 用鼠标选中文本，被选中的文本会被自动复制到tmux的剪贴板

(3) 按下Ctrl + b后松开手指，然后按]，会将剪贴板中的内容粘贴到光标处

(19) tmux卡死的时候：按住Ctrl键，再按五次x键（强行终断传输）

三、vim教程

功能：

(1) 命令行模式下的文本编辑器。

(2) 根据文件扩展名自动判别编程语言。支持代码缩进、代码高亮等功能。

(3) 使用方式：vim filename

如果已有该文件，则打开它。

如果没有该文件，则打开个一个新的文件，并命名为filename

模式：

(1) 一般命令模式

默认模式。命令输入方式：类似于打游戏放技能，按不同字符，即可进行不同操作。可以复制、粘贴、删除文本等。

(2) 编辑模式

在一般命令模式里按下i，会进入编辑模式。

按下ESC会退出编辑模式，返回到一般命令模式。

(3) 命令行模式

在一般命令模式里按下:/?三个字母中的任意一个，会进入命令行模式。命令行在最下面。

可以查找、替换、保存、退出、配置编辑器等。

操作：

(1) i：进入编辑模式

(2) ESC：进入一般命令模式

(3) h 或 左箭头键：光标向左移动一个字符

(4) j 或 向下箭头：光标向下移动一个字符

(5) k 或 向上箭头：光标向上移动一个字符

(6) l 或 向右箭头：光标向右移动一个字符

(7) n<Space>：n表示数字，按下数字后再按空格，光标会向右移动这一行的n个字符

(8) 0 或 功能键[Home]：光标移动到本行开头

(9) $ 或 功能键[End]：光标移动到本行末尾

(10) G：光标移动到最后一行
(11) :n 或 nG：n为数字，光标移动到第n行
(12) gg：光标移动到第一行，相当于1G
(13) n<Enter>：n为数字，光标向下移动n行
(14) /word：向光标之下寻找第一个值为word的字符串。
(15) ?word：向光标之上寻找第一个值为word的字符串。
(16) n：重复前一个查找操作
(17) N：反向重复前一个查找操作
(18) :n1,n2s/word1/word2/g：n1与n2为数字，在第n1行与n2行之间寻找word1这个字符串，并将该字符串替换为word2
(19) :1,$s/word1/word2/g：将全文的word1替换为word2
(20) :1,$s/word1/word2/gc：将全文的word1替换为word2，且在替换前要求用户确认。
(21) v：选中文本
(22) d：删除选中的文本
(23) dd：删除当前行
(24) y：复制选中的文本
(25) yy：复制当前行
(26) p：将复制的数据在光标的下一行/下一个位置粘贴
(27) u：撤销
(28) Ctrl + r：取消撤销
(29) 大于号 >：将选中的文本整体向右缩进一次
(30) 小于号 <：将选中的文本整体向左缩进一次
(31) :w 保存
(32) :w! 强制保存
(33) :q 退出
(34) :q! 强制退出
(35) :wq 保存并退出
(36) :set paste 设置成粘贴模式，取消代码自动缩进
(37) :set nopaste 取消粘贴模式，开启代码自动缩进
(38) :set nu 显示行号
(39) :set nonu 隐藏行号
(40) gg=G：将全文代码格式化
(41) :noh 关闭查找关键词高亮
(42) Ctrl + q：当vim卡死时，可以取消当前正在执行的命令

异常处理：

每次用vim编辑文件时，会自动创建一个.filename.swp的临时文件。

如果打开某个文件时，该文件的swp文件已存在，则会报错。此时解决办法有两种：

(1) 找到正在打开该文件的程序，并退出
(2) 直接删掉该swp文件即可

四、shell语法

Shell教程

五、ssh

1. ssh登录

基本用法

远程登陆服务器：

```
ssh user@hostname
```

- `user`：用户名
- `hostname`：IP地址或域名

第一次登陆时会提示：

```
The authenticity of host '123.57.47.211 (123.57.47.211)' can't be established.
ECDSA key fingerprint is SHA256:iy237yysfCe013/l+kpDGfEG9xxHxm0dnxnAbJTPpG8.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

输入`yes`，然后回车即可。

这样会将该服务器的信息记录在`~/.ssh/known_hosts`文件中。

然后输入密码即可登陆到远程服务器中。

默认登陆端口号为22。如果想登陆某一特定端口：

```
ssh user@hostname -p6000
```

配置文件

创建文件`~/.ssh/config`

然后在文件中输入：

```
Host myserver1
    HostName IP地址或域名
    User 用户名


Host myserver2
    HostName IP地址或域名
    User 用户名
    Port 端口号
```

之后再使用服务器时，可以直接使用别名`myserver1`，`myserver2`。

密钥登陆

创建密钥：

```
ssh-keygen
```

然后一直回车即可。

执行结束后，`~/.ssh/`目录下会多两个文件：

- `id_rsa`：私钥
- `id_rsa.pub`：公钥

之后想免密登陆哪个服务器，就将公钥传给哪个服务器即可。

例如，想免密登陆`myserver`服务器。则将公钥中的内容，复制到`myserver`中的`~/.ssh/authorized_keys`文件里即可。

也可以使用如下命令一键添加公钥：

```
ssh-copy-id myserver
```

执行命令

命令格式：

```
ssh user@hostname command
```

例如：

```
ssh user@hostname ls -a
```

或者

```
# 单引号中的 $i 可以求值
ssh myserver 'for ((i = 0; i  < 10; i ++)) do cheo $i; done'
```

或者

```
# 双引号中的 $i 不可以求值
ssh myserver "for ((i=0; i < 10; i ++)) do echo $i; done"
```

2. scp传文件

基本用法
命令格式：

```
scp source destination
```

将`source`路径下的文件复制到`destination`中

一次复制多个文件：

```
scp source1 source2 destination
```

复制文件夹：

```
scp -r ~/tmp myserver:/home/acs/
```

将本地家目录中的`tmp`文件夹复制到`myserver`服务器中的`/home/acs/`目录下。

```
scp -r ~/tmp myserver:homework/
```

将本地家目录中的`tmp`文件夹复制到`myserver`服务器中的`~/homework/`目录下。

```
scp -r myserver:homework .
```

将`myserver`服务器中的`~/homework/`文件夹复制到本地的当前路径下。

指定服务器的端口号：

```
scp -P 22 source1 source2 destination
```

注意： `scp`的`-r -P`等参数尽量加在`source`和`destination`之前。

使用`scp`配置其他服务器的`vim`和`tmux`

```
scp ~/.vimrc ~/.tmux.conf myserver:
```

3. xshell中传文件：rz、sz

六、git

1. 代码托管平台

git.acwing.com

2. git基本概念

工作区：仓库的目录。工作区是独立于各个分支的。
暂存区：数据暂时存放的区域，类似于工作区写入版本库前的缓存区。暂存区是独立于各个分支的。
版本库：存放所有已经提交到本地仓库的代码版本
版本结构：树结构，树中每个节点代表一个代码版本。

3. git常用命令

- `git config --global user.name xxx`：设置全局用户名，信息记录在`~/.gitconfig`文件中
- `git config --global user.email xxx@xxx.com`：设置全局邮箱地址，信息记录在`~/.gitconfig`文　件中
- `git init`：将当前目录配置成git仓库，信息记录在隐藏的`.git`文件夹中
- `git add XX`：将XX文件添加到暂存区
- `git add .`：将所有待加入暂存区的文件加入暂存区
- `git rm --cached XX`：将文件从仓库索引目录中删掉
- `git commit -m` "给自己看的备注信息"：将暂存区的内容提交到当前分支
- `git status`：查看仓库状态
- `git diff XX`：查看XX文件相对于暂存区修改了哪些内容
- `git log --pretty=oneline`：查看当前分支的所有版本
- `git reflog`：查看HEAD指针的移动历史（包括被回滚的版本）
- `git reset --hard HEAD^` 或 `git reset --hard HEAD~`：将代码库回滚到上一个版本
- `git reset --hard HEAD^^`：往上回滚两次，以此类推
- `git reset --hard HEAD~100`：往上回滚100个版本
- `git reset --hard` 版本号：回滚到某一特定版本

- `git checkout -- xx` 或 `git restore xx`：将XX文件尚未加入暂存区的修改全部撤销
- `git remote add origin git@git.acwing.com:xxx/XXX.git`：将本地仓库关联到远程仓库
- `git push -u origin master`(第一次需要-u以后不需要)：将当前分支推送到远程仓库
- `git push origin branch_name`：将本地的某个分支推送到远程仓库
- `git clone git@git.acwing.com:xxx/XXX.git`：将远程仓库XXX下载到当前目录下
- `git checkout -b branch_name`：创建并切换到branch_name这个分支
- `git branch`：查看所有分支和当前所处分支
- `git checkout branch_name`：切换到branch_name这个分支
- `git merge branch_name`：将分支branch_name合并到当前分支上
- `git branch -d branch_name`：删除本地仓库的branch_name分支
- `git branch branch_name`：创建新分支
- `git push --set-upstream origin branch_name`：设置本地的branch_name分支对应远程仓库的

  branch_name分支
- `git push -d origin branch_name`：删除远程仓库的branch_name分支
- `git pull`：将远程仓库的当前分支与本地仓库的当前分支合并
- `git pull origin branch_name`：将远程仓库的branch_name分支与本地仓库的当前分支合并
- `git branch --set-upstream-to=origin/branch_name1 branch_name2`：将远程的branch_name1分支与本地的branch_name2分支对应
- `git checkout -t origin/branch_name` 将远程的branch_name分支拉取到本地
- `git stash`：将工作区和暂存区中尚未提交的修改存入栈中
- `git stash apply`：将栈顶存储的修改恢复到当前分支，但不删除栈顶元素
- `git stash drop`：删除栈顶存储的修改
- `git stash pop`：将栈顶存储的修改恢复到当前分支，同时删除栈顶元素
- `git stash list`：查看栈中所有元素

七、管道、环境变量与常用命令

1. 管道

概念

管道类似于文件重定向，可以将前一个命令的 `stdout` 重定向到下一个命令的 `stdin`。

要点

1. 管道命令处理 `stdout`，会忽略 `stdin`。
2. 管道右边的命令必须能接受 `stdin`。
3. 多个管道命令可以串联。

与文件重定向的区别

1. 文件重定向左边为命令，右边为文件。
2. 管道左右两边均为命令，左边有 `stdout`，右边有 `stdin`。

举例
统计当前目录下所有的python文件的总行数，其中`find`、`xargs`、`wc`等命令可以参考下面常用命令的内容。

```
find . -name '*.py' | xargs cat | wc -l
```

2. 环境变量

概念
Linux系统中会用很多环境变量来记录配置信息。
环境变量类似于全局变量，可以被各个进程访问到。我们可以通过修改环境变量来方便地修改系统配置。

查看
列出当前环境下的所有环境变量：

```
env     # 显示当前用户的变量
set     # 显示当前shell的变量，包括当前用户的变量;
export  # 显示当前导出成用户变量的shell变量

echo $PATH # 输出某个环境变量的值
```

修改
环境变量的定义、修改、删除操作可以参考3. shell语法——变量这一节的内容。
为了将对环境变量的修改应用到未来所有环境下，可以将修改命令放`~/.bashrc`文件中。
修改完`~/.bashrc`文件后，记得执行`source ~/.bashrc`，来将修改应用到当前的bash环境下。

为何将修改命令放到`~/.bashrc`，就可以确保修改会影响未来所有的环境呢？

- 每次启动bash，都会先执行`~/.bashrc`。
- 每次ssh登陆远程服务器，都会启动一个bash命令行给我们。
- 每次tmux新开一个pane，都会启动一个bash命令行给我们。
- 所以未来所有新开的环境都会加载我们修改的内容。

常见环境变量

1. HOME：用户的家目录。
2. PATH：可执行文件（命令）的存储路径。路径与路径之间用:分隔。当某个可执行文件同时出现在多个路径中时，会选择从左到右数第一个路径中的执行。下列所有存储路径的环境变量，均采用从左到右的优先顺序。
3. LD_LIBRARY_PATH：用于指定动态链接库(.so文件)的路径，其内容是以冒号分隔的路径列表。
4. C_INCLUDE_PATH：C语言的头文件路径，内容是以冒号分隔的路径列表。
5. CPLUS_INCLUDE_PATH：CPP的头文件路径，内容是以冒号分隔的路径列表。
6. PYTHONPATH：Python导入包的路径，内容是以冒号分隔的路径列表。
7. JAVA_HOME：jdk的安装目录。
8. CLASSPATH：存放Java导入类的路径，内容是以冒号分隔的路径列表。

3. 常用命令

系统状况

1. top：查看所有进程的信息（Linux的任务管理器）
    - 打开后，输入M：按使用内存排序
    - 打开后，输入P：按使用CPU排序
    - 打开后，输入q：退出
2. df -h：查看硬盘使用情况
3. free -h：查看内存使用情况
4. du -sh：查看当前目录占用的硬盘空间
5. ps aux：查看所有进程
6. kill -9 pid：杀死编号为pid的进程
    - 传递某个具体的信号：kill -s SIGTERM pid
7. netstat -nt：查看所有网络连接
8. w：列出当前登陆的用户
9. ping www.baidu.com：检查是否连网

文件权限

1. chmod：修改文件权限
    - chmod +x xxx：给xxx添加可执行权限
    - chmod -x xxx：去掉xxx的可执行权限
    - chmod 777 xxx：将xxx的权限改成777
    - chmod 777 xxx -R：递归修改整个文件夹的权限

文件检索

1. `find /path/to/directory/ -name '*.py'`：搜索某个文件路径下的所有*.py文件
2. `grep xxx`：从stdin中读入若干行数据，如果某行中包含xxx，则输出该行；否则忽略该行。
3. `wc`：统计行数、单词数、字节数
    - 既可以从stdin中直接读入内容；也可以在命令行参数中传入文件名列表；
    - `wc -l`：统计行数
    - `wc -w`：统计单词数
    - `wc -c`：统计字节数

4. `tree`：展示当前目录的文件结构
    - `tree /path/to/directory/`：展示某个目录的文件结构
    - `tree -a`：展示隐藏文件
5. `ag xxx`：搜索当前目录下的所有文件，检索xxx字符串
6. `cut`：分割一行内容
    - 从stdin中读入多行数据
    - `echo $PATH | cut -d ':' -f 3,5`：输出PATH用:分割后第3、5列数据
    - `echo $PATH | cut -d ':' -f 3-5`：输出PATH用:分割后第3-5列数据

- `echo $PATH | cut -c 3,5`：输出PATH的第3、5个字符
- `echo $PATH | cut -c 3-5`：输出PATH的第3-5个字符

7. `sort`：将每行内容按字典序排序
   - 可以从stdin中读取多行数据
   - 可以从命令行参数中读取文件名列表
8. `xargs`：将stdin中的数据用空格或回车分割成命令行参数
   - `find . -name '*.py' | xargs cat | wc -l`：统计当前目录下所有python文件的总行数

查看文件内容

1. `more`：浏览文件内容
   - 回车：下一行
   - 空格：下一页
   - b：上一页
   - q：退出
2. `less`：与more类似，功能更全
   - 回车：下一行
   - y：上一行
   - Page Down：下一页
   - Page Up：上一页
   - q：退出
3. `head -3 xxx`：展示xxx的前3行内容
   - 同时支持从stdin读入内容

4. `tail -3 xxx`：展示xxx末尾3行内容
   - 同时支持从stdin读入内容

用户相关

1. `history`：展示当前用户的历史操作。内容存放在~/.bash_history中

工具

1. `md5sum`：计算md5哈希值
   - 可以从stdin读入内容
   - 也可以在命令行参数中传入文件名列表；

2. `time command`：统计command命令的执行时间
3. `ipython3`：交互式python3环境。可以当做计算器，或者批量管理文件。
   - `! echo "Hello World"`：!表示执行shell脚本
4. `watch -n 0.1 command`：每0.1秒执行一次command命令
5. `tar`：压缩文件
   - `tar -zcvf xxx.tar.gz /path/to/file/*`：压缩
   - `tar -zxvf xxx.tar.gz`：解压缩

6. `diff xxx yyy`：查找文件xxx与yyy的不同点

安装软件

1. sudo command：以root身份执行command命令
2. apt-get install xxx：安装软件
3. pip install xxx --user --upgrade：安装python包

4. Filter summary

- cat - 用于连接文件并打印到标准输出设备上
- head - 查看文件的开头部分的内容，有一个常用的参数 **-n** 用于显示行数，默认为 10
- tail - 查看文件的内容，有一个常用的参数 **-f** 常用于查阅正在改变的日志文件。
- grep - 查找某文件的内容符合所指定的范本样式，grep 指令会把含有范本样式的那一列显示出来。
- sed - 依照脚本的指令来处理、编辑文本文件。
- uniq - 用于检查及删除文本文件中重复出现的行列，一般与 sort 命令结合使用。
- cut - 用于显示每行从开头算起 num1 到 num2 的文字。
- tr - 从标准输入设备读取数据，经过字符串转译后，将结果输出到标准输出设备。
- wc - wc命令用于计算字数。
- paste - 用于合并文件的列。
- join - 将两个文件中，指定栏位内容相同的行连接起来。
- sort - 用于将文本文件内容加以排序。
- more - 类似 cat ，不过会以一页一页的形式显示，更方便使用者逐页阅读
- less - 可以随意浏览文件，支持翻页和搜索，支持向上翻页和向下翻页。
- find - 用来在指定目录下查找文件。任何位于参数之前的字符串都将被视为欲查找的目录名。
- xargs - 默认的命令是 echo，这意味着通过管道传递给 xargs 的输入将会包含换行和空白，不过通过 xargs 的处理，换行和空白将被空格取代。
- tee - 从标准输入设备读取数据，将其内容输出到标准输出设备，同时保存成文件。

1. 水平切片 - 选择行的子集： cat、head 、tail、grep、sed、uniq
2. 垂直切分--选择列的子集： cut, sed
3. 替换: tr, sed
4. 汇总，简单统计：wc, uniq
5. 组装 - 结合数据源： paste, join
6. 重新排序: sort
7. 查看（总是在管道的末端）： more, less
8. 文件系统过滤：find
9. 可编程过滤器: sed, (and perl)

八、docker教程

1. 将当前用户添加到`docker`用户组

为了避免每次使用`docker`命令都需要加上sudo权限，可以将当前用户加入安装中自动创建的`docker`用户组(可以参考官方文档)：

```
sudo usermod -aG docker $USER
```

2. 镜像（images）

1. docker pull ubuntu:20.04：拉取一个镜像
2. docker images：列出本地所有镜像
3. docker image rm ubuntu:20.04 或 docker rmi ubuntu:20.04：删除镜像ubuntu:20.04
4. docker [container] commit CONTAINER IMAGE_NAME:TAG：创建某个container的镜像
5. docker save -o ubuntu_20_04.tar ubuntu:20.04：将镜像ubuntu:20.04导出到本地文件ubuntu_20_04.tar中
6. docker load -i ubuntu_20_04.tar：将镜像ubuntu:20.04从本地文件ubuntu_20_04.tar中加载出来

3. 容器(container)

1. docker [container] create -it ubuntu:20.04：利用镜像ubuntu:20.04创建一个容器。
2. docker ps -a：查看本地的所有容器
3. docker [container] start CONTAINER：启动容器
4. docker [container] stop CONTAINER：停止容器
5. docker [container] restart CONTAINER：重启容器
6. docker [contaienr] run -itd ubuntu:20.04：创建并启动一个容器
7. docker [container] attach CONTAINER：进入容器
    - 先按Ctrl-p，再按Ctrl-q可以挂起容器
8. docker [container] exec CONTAINER COMMAND：在容器中执行命令
9. docker [container] rm CONTAINER：删除容器
10. docker container prune：删除所有已停止的容器
11. docker export -o xxx.tar CONTAINER：将容器CONTAINER导出到本地文件xxx.tar中
12. docker import xxx.tar image_name:tag：将本地文件xxx.tar导入成镜像，并将镜像命名为image_name:tag
13. docker export/import与docker save/load的区别：
    - export/import会丢弃历史记录和元数据信息，仅保存容器当时的快照状态
    - save/load会保存完整记录，体积更大
14. docker top CONTAINER：查看某个容器内的所有进程
15. docker stats：查看所有容器的统计信息，包括CPU、内存、存储、网络等信息
16. docker cp xxx CONTAINER:xxx 或 docker cp CONTAINER:xxx xxx：在本地和容器间复制文件
17. docker rename CONTAINER1 CONTAINER2：重命名容器
18. docker update CONTAINER --memory 500MB：修改容器限制

4. 实战

进入AC Terminal，然后：

```
scp /var/lib/acwing/docker/images/docker_lesson_1_0.tar server_name:  # 将镜像上传到自己租的云端服务器
ssh server_name  # 登录自己的云端服务器

docker load -i docker_lesson_1_0.tar  # 将镜像加载到本地
docker run -p 20000:22 --name my_docker_server -itd docker_lesson:1.0  # 创建并运行docker_lesson:1.0镜像

docker attach my_docker_server  # 进入创建的docker容器
passwd  # 设置root密码
```

去云平台控制台中修改安全组配置，放行端口`20000`。

返回AC Terminal，即可通过`ssh`登录自己的`docker`容器：

```
ssh root@xxx.xxx.xxx.xxx -p 20000  # 将xxx.xxx.xxx.xxx替换成自己租的服务器的IP地址
```

然后，可以仿照上节课内容，创建工作账户`acs`。

最后，登录配置`docker`容器的别名和免密登录。
如果`apt-get`下载软件速度较慢，可以参考清华大学开源软件镜像站中的内容，修改软件源。